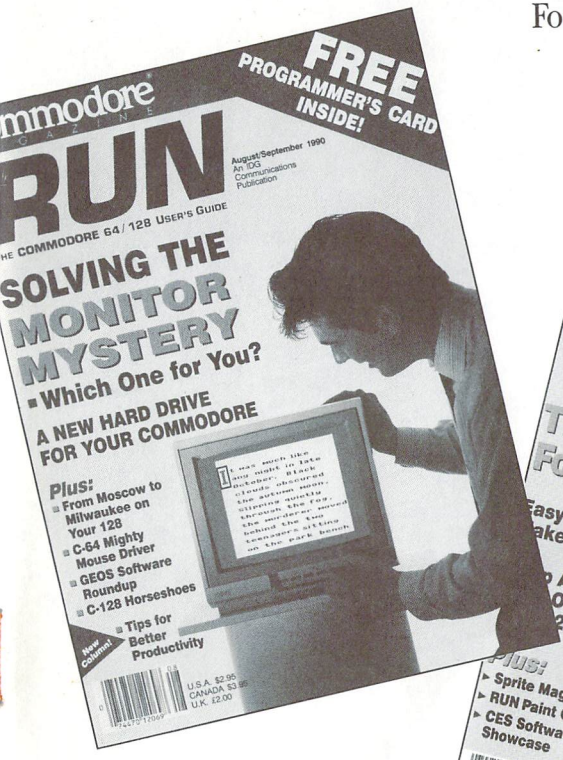
August–September–October 1990 Edition

RE-RUN

RUN Programs on Disk

For the C-64 and C-128



Plus: Extra Bonus Programs!

Introduction

August–September–October '90 ReRUN

THE ENTIRE *RUN* EDITORIAL STAFF takes special pride in presenting this August/September/October edition of ReRUN. We've assembled a most impressive and creative lineup of programs, and this promises to be one of the best ReRUN packages of 1990.

We kick off with How Far Is It To. . .?. This C-128, 80-Column mode program lets you quickly find the distance between most major cities around the world. Best of all, we were able to include numerous cities on this disk-based version that, due to space limitations, were not listed in the magazine.

Exercise Your Mouscles, a C-64 mouse-driver written by veteran programmer Jim Borden, is next on the list. Jim worked long and hard developing the best 1531 mouse driver we've seen yet for the C-64. A demo program is also included to help you put the driver to use.

After you've finished using the mouse driver, check out Pop-Top!. Exciting joystick-controlled action from game programmer Tony Brantner, the object of Pop-Top! is to burst balloons as they drift by. It's great fun for youngsters of all ages.

Horseshoes respresents our 128 Mode program for the August/September 1990 issue of *RUN*. This humorous two-player game pits formidable Dolly against dead-ringer Guy in a 40-Column mode horseshoe showdown.

The October issue of *RUN* is well represented with A Notable Basic, one of the best programs published this year. It gives your C-64 every music command (with the exception of Sound) that's available through the C-128's Basic 7.0. An elaborate musical composition is also included to demonstrate the incredible music-making ability SID Basic 64 gives your Commodore. It even takes advantage of extra voices, in case your C-64 has a second SID (Sound Interface Device) chip installed.

Time Clock is also from *RUN*'s October issue. While digital clock programs have been available since the introduction of the C-64, Time Clock breaks tradition by offering an on-screen analog clock, complete with a second hand, for both 64 and 128 modes. The 128

version, incidentally, displays the clock in 40-Column mode only. Best of all, simple Basic programs won't disrupt the operation of either version of Time Clock!

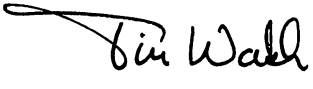
RUN Paint Renamer, written by *RUN*'s long-time contributor, Hugh McMenamin, converts Doodle! and Koala graphics into RUN Paint-compatible screens. Sprite Magician, a C-64 sprite animator and controlling program, brings 11 new sprite commands to Basic 2.0 to ease the otherwise troublesome task of moving sprites around the screen. An exciting demo program is also included.

Device Toggler, a C-128 device number switching utility, represents the October 128 Mode program. Use it to switch device numbers without fiddling with your drive's DIP switches.

And last, but certainly not least, we've included two bonus programs. For the kids, there's Apple Harvest, a program appropriate for autumn. The object of the game is to gather apples as they fall from trees, while you avoid getting beaned, or in this case, "appled" by the apples your basket misses.

When the kids are done gathering apples, Mom and Dad will appreciate Disk Directory Organizer, another Jim Borden original. The beauty of this ingenious disk cataloging program is that multiple disks can be catagorized, sorted and then printed on a single sheet of paper! Accomplished through the Compressed mode of your Epson-compatible printer, five columns of directory listings are printed across each sheet. Disk Directory Organizer is for anyone looking for a useful, easy-to-use disk directory printing utility.

I look forward to coming back soon with another exciting and productive lineup of programs—just in time for the holiday season. Until then, happy computing!



Technical Manager
RUN *Magazine*

Directory

* — C-128 mode only
£ — Bonus program

Before you run a program, carefully read the documentation that pertains to it.

How To Load

Loading from Menu

To get started, C-64 users should type LOAD "MENU 64",8 and press the return key. When you get the Ready prompt, the menu is loaded and you should type RUN to see a list of the programs on your disk. C-128 users need only press the shift and run-stop keys. When all the programs are displayed on the screen, you can run the one you select by pressing a single key.

Loading from Keyboard

If you do not wish to use the menu program, follow these instructions.

C-64: To load a C-64 program written in Basic, type: LOAD "DISK FILENAME",8 and then press the return key. The drive will whir while the screen prints LOADING and then READY, with a flashing cursor beneath. Type RUN and press the return key. The program will then start running. To load a C-64 program written in machine language (ML), type: LOAD "DISK FILENAME",8,1

C-128: All C-64 programs can be run on the C-128 as long as your computer is in C-64 mode. All C-128 programs are clearly labeled on the directory page. Your C-128 *must* be in C-128 mode to run these programs. To load a C-128 mode program, press the F2 key, type the disk filename and then press the return key. When the program has loaded, type RUN.

Making Copies of ReRUN Files

Many programs on your ReRUN disk have routines that require a separate disk onto which the program writes or saves subfiles. To use these programs, you must first make a copy of the original program onto another disk that has enough free space on it to hold these newly written subfiles.

It's simple to make a copy of a Basic program. Just load it into your computer as outlined above, and then save the program back onto a separate disk that has plenty of free space for extra files.

Copying an ML program is not so simple. You cannot simply load and save an ML program; you'll need to use a disk-backup utility program, such as the one on your Commodore Test Demo disk.

RUN it right: **C-128 (in 80-Column mode)**

How Far Is It To. . .?

By Larry Pankey

THE MAY 1988 ISSUE of *Commodore Magazine* contained a type-in Basic program by Leo W. Brenneman entitled Air Miles. Written for the C-64, it calculated great circle distances between cities around the world. The many city names and their latitude and longitude (lat/lon) coordinates were contained in Data statements.

Air Miles 128 is an 80-column enhanced version of the original that takes advantage of the C-128's Fast mode. It also increases program speed by searching the Data statements, instead of reading the data into memory as strings.

After loading Air Miles 128, using Menu 128, the program first displays a main menu screen that includes a brief description of Air Miles 128 and its five main menu options.

Selecting option 1, Directory of Cities, brings up a submenu for viewing a list of all cities in the Data statements or just those beginning with a particular letter. (Unlike the magazine version, we've included a large number of cities in the Data statements here.)

When you select main menu option 2, Distance Between Cities, the program asks for the names of two cities. If you type a name that's not in the data, or spell it differently than in the data, the program displays a "Not in List. Check Menu Option 1" message. Then, after three seconds, it asks for the city again. Pressing the ↑ (up-arrow) key at any time restores the main menu.

If the program finds the two cities, it displays the distance between them in statute miles, nautical miles and kilometers. Then you can press F7 to return to the main menu or F1 to type in another pair of cities. To repeat a city from the last pair, just press the ← (left-arrow) key, and the name appears.

Option 3, Enter Latitudes and Longitudes, lets you type in coordinates for two locations of your own choosing, called Point A and Point B. F7 returns you to the main menu; F1 lets you continue.

Option 4, Print Hard Copy of Cities, is similar to option 1, except that all city data is sent to your printer.

Option 5, Quit, provides an exit from Air Miles 128 to Basic.

The program allows 22 characters for a city name. If you try to enter more characters than are allowed at this, or any, prompt, the screen flashes red, and the speaker beeps. Air Miles 128 also has a trap routine to catch syntax errors.

A Geography Lesson

The location of any point on the earth can be identified by its latitude, or distance north or south of the equator, and its longitude, or distance east or west of Greenwich, England. Latitudes run from 0 degrees at the equator to 90 degrees, north or south, at the poles. The northern hemisphere lies north of the equator, the southern hemisphere south of it. Latitude lines on maps, connecting all points of like latitude, are called parallels.

Longitude lines are called meridians and run north and south from pole to pole. The eastern and western hemispheres are defined by the 180th meridian, on the opposite side of the world from the Greenwich meridian. The International Date Line runs, with a few deviations, along the 180th meridian.

Latitude and longitude are usually stated in degrees, minutes ($^1/_{60}$ of a degree) and, sometimes, seconds ($^1/_{60}$ of a minute). Air Miles 128 doesn't use seconds, since one second latitude represents only about 100 feet.

Adding Data

The Data statements are grouped according to first letter of city name, those names beginning with A in lines 1000–1090, those beginning with B in lines 2000–2110, and so on. If you enter additional cities of your own choosing, be sure to follow this scheme, and don't renumber the program. The data search method relies on cities being separated in this fashion. The line

DATA {UP ARROW},,,,,,

must always be the last line in any alphabetical group. However, you can change that line's number—for example, in this listing, line 1998 can become line 1999 as long as it remains the last line in the "A" group.

To add a city to the data, insert a new line number within the proper group, then type the word DATA followed by the city name, degrees latitude, minutes latitude, N or S, degrees longitude, minutes longitude, and E or W. If you don't know the minutes, type 0.

Each Data line must contain six commas. If you end up with more than 100 cities in an alphabetical group, change the number in the DIMC$(101) statement in line 3 to your new quantity plus 1.

Look in a world atlas for major cities' coordinates. When you have identically named cities in different states or countries, add the state or country name—it can be abbreviated—to one or both. However, don't put a comma between the city name and the state or country.

Mariners and fishermen can find coordinates using oceanographic charts. Of course, you won't have ready-made names for places on the water; you'll have to be creative. For instance, you might name a special fishing spot Bank 207 or Bank 14, for the depth in fathoms. The spot would then go into the B group of Data statements. Use your imagination in naming other places you want to include in the program.

RUN it right: **C-64; Commodore-compatible mouse**

Exercise Your Mouscles

By Jim Borden

WHEN I NEEDED A flexible mouse driver for my C-64, I wrote one to do the job. Called Mouse.P1, it works with a mouse plugged into port 1, and offers several features that make it easy to use from Basic.

The mouse sprite (I used the corners of the normal cursor) is located in a window on the screen. The default window is the full screen, but five memory locations, 990–994, let you move the window almost anywhere. Mouse.P1 also lets you get both the row and column of the mouse sprite and the status of either button with Peeks from Basic.

The locations that pass information back and forth between Basic and Mouse.P1 are listed in Table 1. Note that the maximum X position is a two-byte value. Use the formulas shown for the Poke values, where R is the row number you want, 1–25, and C is the column number, 1–40. The Pokes are used only to change the window to limit the mouse sprite's movement.

Run MOUSE DEMO, using Menu 64, to see the mouse sprite in action. MOUSE DEMO is a three-part demo program that illustrates how to use Mouse.P1.

THE DEMO

Load the demo program, then run it. First it loads Mouse.P1, and then you activate it by entering SYS 49152. At that point, the sprite appears on the screen.

Part of the machine language code defines a sprite for the mouse. If you want another shape for the sprite, you must activate Mouse.P1 first and then change the sprite. Poke the data for the new shape into locations 921–939 in steps of three.

The next lines of the demo set up the variables for the Peek addresses, so it's easy to remember what to peek. For example, if you want to get the row number of the mouse sprite, just use PEEK (RN). The demo gives examples of all the Peeks and Pokes.

As I mentioned, the demo program is actually three demos in one. The first is a very simple lo-res drawing program, with a working part that contains only five lines of Basic code. The other lines are REM and Print statements.

First the drawing program waits to detect a mouse button being pressed. If both buttons are down, the program clears the screen and jumps to the next part of the demo. Line 110 calculates the screen position to peek or poke, then checks to see if the shift/clear-home key combination is being pressed. If line 110 finds those keys down, it clears the screen for another drawing. Line 120 handles the left button. If it's down, the character at location SP is erased to normal video. Line 130 handles the right button by changing the character to reverse video (or turning it on).

As the program is written now, the screen contains only spaces or reverse spaces (and the characters on the original screen until they're cleared). However, two ten-year-old girls "tested" the program for over two hours, so, even though it's simple, it's fun.

The second section of the demo shows the use of a menu (or a multiple-choice question in this case). Here the column isn't required, so only the row is checked. If the row is within the range of answers and either button is down, the chosen answer is highlighted. Line 250 tells you if the answer is incorrect. The correct answer is then shown, and a message is printed telling you to release the button. Next, line 270 waits until both buttons are up. Finally, lines 280 and 290 wait for a button press before moving on to the last

part of the demo. This is necessary so you can see if your answer was correct.

The last part of the demo shows how to read moves for a game such as tic-tac-toe. Lines 370–390 set up a window the same size as the game board, and lines 470 and 480 reset the window to the full screen. The program *doesn't* play the game; it just shows how to read the row and column to get a move. It also rounds off the move to the center of the chosen box. This method can be helpful if you want to use a large board like the one shown in games of your own design.

I've found one minor glitch in the demo that I can't get rid of. To shift F1, Z, C, B, M or space, you must use the *left* shift key. The right one won't work with these keys most of the time.

With Mouse.P1, it's simple to read the status and position of the mouse (as a row and column) and to set up custom window limits for the mouse sprite. Use it—writing a mouse into your Basic programs can be a lot easier than you think.

Table 1. Locations that pass information between Basic and Mouse.P1.

POKE 990,(8*R+34)	Minimum Y position (row) of the mouse
POKE 991,(8*R+34)	Maximum Y position of the mouse
POKE 992,(8*C+8)	Minimum X position (column) of the mouse
POKE 993,((8*C+8) AND 127)	Maximum X position (low byte) of the mouse
POKE 994,-((8*C+8)>255)	Maximum X position (high byte) of the mouse
PEEK (RN)	Row number of the mouse sprite (1–25)
PEEK (CN)	Column number of the mouse sprite (1–40)
PEEK (LB)	Left button status (0=up; 1=down)
PEEK (RB)	Right button status

RUN it right: **C-64; joystick**

Pop-Top!

By Tony Brantner

CHILDREN REALLY GET a bang out of Pop-Top, a colorful C-64 game that combines arcade action with a nonviolent theme. The star of Pop-Top is Topper the Clown. You move Topper back and forth across the bottom of the screen by using a joystick plugged into port 2. Press the firebutton, and Topper's hat pops off his head, zips to the top of the screen and drops down to make a pinpoint landing back on its owner's head.

Your goal in Pop-Top is to make the hat burst the balloons that are floating across the top of the screen. This entails positioning Topper for a good "shot," then moving him to guide the hat's trajectory while airborne. It also means making sure the hat doesn't tangle with a fedora-munching pinwheel that's passing across the screen. The game progresses through five difficulty levels, starting with a single pinwheel and adding another at each additional level.

Topper starts with a wardrobe of three hats. One or two lost are replaced when you hit 20 balloons and move to the next level. If you lose all three, the game ends and resets to level 1.

Your score for each balloon popped is ten multiplied by the current difficulty level. You also receive a bonus of 100 points for each hat remaining as you move to a new level.

The bottom of the screen displays your running score, your highest score for the current game session, the number of hats left at the current level and the number of balloons you must still pop to reach 20 and proceed to the next level.

Just load and run POPTOP, using Menu 64, and start popping those balloons!

RUN it right: **C-128 (in 40-Column mode); two joysticks**

Horseshoes

By Mark Jordan

LET'S TAKE A LOOK at sprites and graphics and combining the two in Horseshoes, a game I wrote for *RUN*'s August/September 128 Mode column. In the program, the Sprite commands create the flying horseshoes and the swinging arms, and the Graphics commands create Guy and Dolly, two limber-armed characters on a colorful background.

Run Horseshoes, using Menu 128, and take a close look at Guy and Dolly. By creating them in two high-resolution sprites (the body plus the head and face), I was able to use three colors in Multicolor mode without sacrificing resolution. Study lines 340–420 to see how I did it.

To create the 28 different arm positions that toss the horseshoes, I would normally generate the needed Data statements, but in this case, it would make the program listing much longer than necessary. Instead, I achieved the same result—and saved a lot of memory—by using the Circle command to make arcs of a circle that I then employed as the arms' different positions. Next, I saved each position to a subscripted variable—ARM$(S,J)—in lines 120–200.

I also designed nine different horseshoe shapes from a single sprite, each shape occupying an eight-by-seven pixel area. By utilizing the SShape, GShape and SPRSAV commands (lines 60–110), each of the nine shapes was derived from the original and transferred to a string variable, HS$(X). So, when the shoe is tossed (lines 690–730), simply cycling through line 720 (the SPRSAV command and the subscript) gives the illusion of a rotating horseshoe.

You can examine the individual parts that make up Guy and Dolly by adding line 55 SPRDEF:STOP to the listing. (Remember to remove this line before playing a game.) The sprite screen will appear when you run the program. Press any number from 2 to 8 to view a sprite. To view another sprite, first erase the previous number with the run-stop key, then enter a new number. To watch the Graphics commands in action, delete the Fast command in line 20.

Horseshoes takes a little over 20 seconds to load in the graphics. It is played with two joysticks—port 1 controls Guy and port 2, Dolly. When you're ready to toss, move the joystick left and right to control arm movement, and press the firebutton to release the shoe. With the right amount of backswing and foreswing, you'll hit at or near the post. Conventional horseshoe scoring is used, with 21 points winning a game. In each round, the player with the best previous score gets the first toss.

RUN it right: **C-64**

A Notable Basic

By Chris Newman and Kent Sullivan

ONE OF THE MOST DISTINCTIVE and powerful features of the C-64 and C-128 is the programmable SID (Sound Interface Device) chip. The SID is responsible for all the sound and music that your C-64/128 makes and has capabilities that far outdistance more expensive computers.

Using the SID on the C-128 is straightforward, because Basic 7.0 supports sound and music creation with six commands: Sound, Play, Envelope, Filter, Tempo and Volume. On the C-64, however, Basic 2.0 forces you to use a litany of Pokes and Peeks to master the magic of SID. Until now, at least. . .

SID Basic 64 brings five of the six Basic 7.0 music commands (all but Sound) to Basic 2.0 in an easy-to-use driver that's compatible with all normal Basic 2.0 programs. It also supports up to six voices (twice the normal) for true stereo music, due to the increasing popularity of adding a second SID chip to the C-64/128 (via an internal modification or a cartridge). In addition, SID Basic 64 can read Basic 7.0 programs that have music commands in them, so you won't have to retype songs that you already have for the C-128.

Starting Notes

After loading SID Basic 64, using Menu 64, try experimenting with each of the music commands. Once you have SID Basic 64

active, reload Menu 64 and run the program called TWINKLE, which requires SID.BASIC to be activated beforehand.

TWINKLE places the melody and counterpoint in voices 1 and 2. In SID Basic 64, voices 1–3 are played through the SID chip inside your C-64/128 (on the *left side* in stereo terminology) while voices 4–6 are played through the second SID chip (on the *right side*), if you have one installed. If you do have a second SID chip, TWINKLE plays a four-voice, stereo song.

COMMANDS

All SID Basic 64 commands work in both Program and Direct modes. Due to space limitations here, we can't go into detail on music theory or the combinations of SID Basic 64 commands, but these commands are well documented in the *Commodore 128 System Guide* and the *Commodore 128 Programmer's Reference Guide*, as well as several other sources.

It's very important to note that values set during execution of SID Basic 64 commands, unlike those set by Basic 7.0 commands, don't get reset, so you need to reload and reenable SID Basic 64 if you wish to restore the default settings. Otherwise, you probably won't get the results you're looking for.

Below is a summary of each of the five commands:

ENVELOPE

Purpose: selects the various ADSR and waveform parameters that create sounds, or "instruments."

Syntax: ENVELOPE e[,a[,d[,s[,r[,wf[,pw]]]]]]
e = envelope number: 0–9 (see Table 1 for predefined values)
a = attack rate: 0–15
d = decay rate: 0–15
s = sustain level: 0–15
r = release rate: 0–15
wf = waveform
  0 = triangle
  1 = sawtooth
  2 = pulse
  3 = noise
  4 = ring modulation
pw = pulse width: 0–4095 (valid only with waveform 2)

Notes: If you wish to use one of the predefined instruments, specify only the e parameter. For example:

ENVELOPE 0

selects the default piano instrument.

If you wish to redefine one of the instruments for your own use, you must specify all the parameters (except pulse width when you're not using the pulse waveform). For example:

ENVELOPE 0,7,9,4,7,0

redefines the default piano instrument as an oboe by giving it an attack of 7, a decay of 9, a sustain of 4, a release of 7 and a triangle waveform.

You can hear what the ten predefined instruments sound like by selecting them with the T parameter of the Play command (below) and specifying a few notes to be played.

Table 1. Predefined (default) envelope values.

Number	Instrument	Attack	Decay
0	Piano	0	9
1	Accordion	12	0
2	Calliope	0	0
3	Drum	0	5
4	Flute	9	4
5	Guitar	0	9
6	Harpsichord	0	9
7	Organ	0	9
8	Trumpet	8	9
9	Xylophone	0	9

FILTER

Purpose: controls the SID filter to alter the sounds that a voice produces.

Syntax: FILTER cf[,lp[,bp[,hp[,res[,chip]]]]]

cf=cutoff frequency: 0–2047
lp=low-pass filter: 0 (default)=off, 1=on
bp=band-pass filter: 0 (default)=off, 1=on
hp=high-pass filter: 0 (default)=off, 1=on
res=resonance level: 0–15
chip=chip number: 0=internal/left, 1=external/right

Notes: You can have any combination of the three filters turned on at one time. The resonance is the "peaking effect" of the sound's

Sustain	Release	Waveform	Width
0	0	2	1536
12	0	1	
25	0	0	
5	0	3	
4	0	0	
2	1	1	
0	0	2	512
9	0	2	2048
4	1	2	512
0	0	0	

frequency as it nears the cutoff frequency. Note that the "chip" parameter is a SID Basic 64 enhancement to support six voices; it is not part of Basic 7.0. If you don't specify a chip, the filters on both SID chips will be affected.

PLAY

Purpose: outputs a string of musical notes to the SID chip, much like Print outputs a string of characters to the screen.

Syntax: PLAY "V<n>,O<n>,T<n>,U<n>,X<n>,<[notes or elements]. . .>"

V<n> = voice number: n = 1–6 (1 is the default)
O<n> = octave number: n = 0–6 (4 is the default)
T<n> = tone envelope number: n = 0–9 (0 is the default)
U<n> = volume level: n = 0–9 (9 is the default)
X<n> = filter: n = 0 or 1 (0 is the default)
notes = valid musical notes: C,D,E,F,G,A,B
elements = modifiers to notes (all but R and M must be followed by a note letter)
W = whole note
H = half note
Q = quarter note
I = eighth note
S = sixteenth note
. = dotted note
R = rest
M = measure (wait for all voices to finish current note)
# = sharp
$ = flat

Notes:
1. SID Basic 64 expands the V parameter to accommodate up to six voices; Basic 7.0 has a range of 1–3.
2. The values for T correspond to the ten instruments defined through the Envelope command.

3. The Volume command (see below) has a range of 0–15, but the U parameter of Play scales this into ten steps, as follows:

U	Volume
0	0
1	1
2	3
3	5
4	7
5	8
6	12
7	13
8	14
9	15

Also, the U part of U<n> is optional (as it is in Basic 7.0). You can specify just the number, and Play will assume you want to change the volume.

4. An X value of 0 means turn the filter off, while a value of 1 means turn the filter on.

5. *Important:* If the Play command doesn't produce any sound, remember to make sure the volume is set to a value other than 0.

An example of the Play command is:

```
PLAY "V1 O4 T7 U8 X0"
```

which sets up voice 1 to play in octave 4 with tone envelope 7 (organ), volume 8 (equal to VOL 14), and the filter off. Another example is:

```
PLAY "V3 O3 I C V2 O5 Q#D"
```

which plays an eighth note C in the third octave using voice 3 and then plays a quarter note D-sharp in the fifth octave using voice 2.

Note that in the two examples above, we inserted spaces between the elements of the strings to improve readability; they aren't mandatory.

TEMPO

Purpose: adjusts the speed at which the music plays.

Syntax: TEMPO r

r = rate of play: 1–255 (255 is fastest, 8 is the default)

Notes: The default tempo of 8 equals M.M. 100 (100 beats per minute). Although all applicable technical notes say that Basic 7.0's default tempo is 8, it's actually 16. We decided to make SID Basic 64 follow the written specifications.

VOLUME

Purpose: adjusts the loudness at which the music plays.

Syntax: VOL l[,c]

l = volume level: 0–15 (15 is loudest and the default)
c = chip number: 0 = internal/left, 1 = external/right

Notes: Although the default volume is 15, a volume of 12 is the loudest you'll normally need; values above 12 can distort some instruments. Note that the c parameter is a SID Basic 64 enhancement to support six voices; it is not part of Basic 7.0. If you don't specify c, the volume for both SID chips will be changed.

Technical Notes

If you have a second SID chip, you might want to "spread out" a three-voice song across both the left and right channels. You can do this by copying the code for voices 1–3 and changing it to use voices 4–6. However, there's an easier way: You can tell SID Basic 64 to play a voice on the second SID chip without having to rewrite the Basic program. Just change the chip the voice is mapped to with:

POKE 49298+<voice number>(1–6),<value>

where value equals 0 for the internal SID (left side) or 1 for the external SID (right side). To spread out a three-voice song, try moving voice 2 to the second chip with POKE 49300,1. Try moving other voices and experimenting with other configurations.

SID Basic 64 assumes that if you have a second SID chip, it's mapped into memory at $DE00. If yours resides at a different location ($DF00 is the other likely candidate), you can tell SID Basic 64 by poking location 49306 with the high byte of the address. For instance, for a chip at $DF00, you'd use POKE 49306,223 (223 = $DF). The location of the internal SID chip is stored in 49305

and is 212 by default (212=$D4). This default value works on all C-64s and C-128s.

As mentioned above, SID Basic 64 can list Basic 7.0 programs. It correctly displays all six sound and music commands and can handle most other Basic 7.0 commands by displaying the hexadecimal value of the Basic keyword (token) within brackets (for example, COLOR becomes [E7]). It does not correctly handle "dual-token functions" and will report a syntax error if it encounters one. This shouldn't be much of a problem, however, since you can easily edit out the nonmusic commands from any Basic 7.0 program you wish to play with SID Basic 64.

SID Basic 64 is compatible with other "wedge" programs that use the "error" or CHRGET techniques, but not those that redirect the IMAIN, ICRNCH, IQPLOP or IGONE vectors (locations $0304–$0309 in memory). It is compatible with the Commodore DOS wedge, but with Creative Micro Design's JiffyDOS, it tries to tokenize some characters incorrectly. For instance, JiffyDOS's /INV* would normally load the first program that begins with INV, but SID Basic 64 converts the * into its tokenized form and causes a File Not Found error. You can get around this effect by adding a leading quote to the line: /"INV*. It is important to remember to always install SID Basic 64 after any other wedges you wish to use.

We wish to thank Art Hunkins for his inspiration and help, and Dan Heeb for the information gleaned from his book *VIC-20 and Commodore 64 Toolkit: Basic*.

RUN it right: **C-64 or C-128 (in 40-Column mode)**

Time Clock

By Terry Bryner

WHEN YOU'VE BEEN READING standard clocks for years, a digital clock just isn't the same, especially when it gets lost in the jumble of letters on a computer screen. Time Clock fixes this problem by putting an old-fashioned round clock, complete with a second hand, on the screen of your C-64 or C-128. The clock keeps accurate

time, and you can read it at a glance while typing in Basic programs, running some of them and performing disk operations.

There are two versions of Time Clock, one for the C-64 and one for the C-128 in 40-Column mode. Load and run either, using Menu 64 or Menu 128.

To use the clock, enter two digits each for the current hour, minute and second at the HHMMSS prompt. When you press the return key, the clock appears in the upper-right corner of the screen, complete with moving second hand, and it continues to run until you deactivate it by pressing the run-stop and restore keys simultaneously.

To load another Basic program, type NEW and press return to clear Time Clock from memory. Because the program erases itself from memory, you can work on other Basic programs while the clock remains active. You can hide the clock with POKE 53269,0 and then restore it with POKE 53269,240.

If you need to move the program to another disk, simply load it, swap disks and save it. Although written in machine language, Time Clock behaves like a normal Basic program.

The C-64 version is designed to operate in the normal, bank 0 Video mode. The C-128 version doesn't load properly after the commands GRAPHIC1:GRAPHIC0 are used to move the start of Basic, and either version displays garbage if the sprite pointers are manipulated. However, for typical extended conversations with the screen editor, the clock is a useful reminder of the fleet wings of time.

RUN it right: **C-128 (in 40-Column mode)**

RUN Paint Renamer

By Hugh McMenamin

RUN PAINT, *RUN*'s powerful drawing and painting program (March 1989), is easy to use and supports all the common graphics formats. However, when importing a file into RUN Paint from another graphics program, you must make the filename compatible with RUN Paint. My Convert to RUN Paint program eliminates this hassle for C-128 users by loading a graphics file into the computer,

then saving it to a destination disk and automatically renaming it in the process.

To use the program, just load and run it, using Menu 128; then insert the source disk, with the files to be renamed, into the drive.

Convert to RUN Paint starts by presenting a menu of file types it can rename: Koala and Doodle!. Enter the first letter of the type you want, and the program responds with a directory of those files on the disk. Now enter the name of the particular file you want to rename, leaving out the prefix or suffix specific to the graphics program that created the file—a reverse video A for Koala or DD for Doodle!. Note that you should keep the PIC in those Koala filenames that contain it; delete only the reverse A. Also, you must type in each filename, even if you want to rename all the files on the disk; the wild-card asterisk won't work.

When you've typed the desired filename and pressed the return key, the program loads the file. Next, insert the destination disk and press return to save the file with its new name. The program terminates when the save is done. To rename another file, run it again.

A new filename consists of the filename you entered from the source disk directory with a prefix of RPM. or RPH. (note the dot following each), for RUN Paint med-res or hi-res, added on. If the filename you entered is longer than 12 characters, it's truncated so that the total won't exceed 16 characters. Also, if the filename you entered duplicates one already on the destination disk or if the destination disk is full, the program reports an error and tells you to press any key to continue.

With your files renamed, you can load them into RUN Paint using the appropriate format—med-res or hi-res—and modify them to your heart's content.

Sprite Magician

By Scott Weisgarber

SPRITE MAGICIAN IS A complete sprite movement and animation controller for the C-64. Using its 11 SYS commands, you can position a sprite anywhere, define interrupt-driven movement at any speed in any direction, create borders that confine individual sprites to certain areas, make sprites wrap around or bounce when they hit a border or the screen edge, and set up automatic animation. All the commands include a parameter for specifying the sprite you want the command to affect. Acceptable values are 1–8, for the eight C-64 sprites.

To see Sprite Magician in action, use Menu 64 to load and run SPRITE DEMO. It uses almost all of Sprite Magician's features.

START-UP

After loading Sprite Magician, you'll see the screen turn black and the logo appear. *Note: This command not only activates the program, but clears the values of other variables, so use it with caution.*

To disable Sprite Magician, enter SYS 52831, which changes the IRQ back to the default hardware vector. To reactivate the program after disabling it, enter SYS 52828, which changes the IRQ back to Sprite Magician without resetting any pointers.

To reset all the pointers—for borders, movement settings, bouncing, and so on—without bothering the operating system, disable Sprite Magician, enter SYS 51466, and then reactivate the program. SYS 51478 resets only the borders.

Because of Sprite Magician's animation feature, you can't poke directly to the regular VIC-II sprite block pointers (addresses 2040–2047). You must go to the new addresses of 53140–53147. For example, instead of entering POKE 2040,192, you'd enter POKE 53140,192.

Advanced programmers should note that changing video banks also relocates the regular 2040–2047 sprite block pointers. To tell Sprite Magician what the new pointer locations should be, enter

SYS 52788, followed by the new location (SYS 52788,4096 to change the new pointers to 4096, for example.)

THE COMMANDS

Position

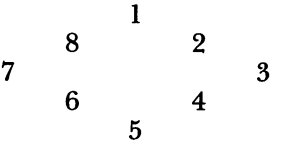
SYS 52834,sprite number,X,Y

X and Y are the screen coordinates where you want the sprite positioned.

Move

SYS 52837,sprite number,direction,X speed,Y speed

Direction values range from 1 to 8, as shown in the following diagram.



The X speed is horizontal, the Y is vertical, and 1 to 149 pixels per second is the acceptable range for both. As an example of the Move command, SYS 52837,1,2,10,5 sends sprite 1 toward the upper right with a horizontal speed of ten pixels per second and a vertical speed of five pixels per second.

Borders

Using borders, you can box individual sprites into certain sections of the screen—useful, say, when writing an arcade game where a creature gets trapped in a cage. When Sprite Magician is initialized, borders for all eight sprites are set at the actual screen border. The following four commands are used to define borders.

Top Border

SYS 52840,sprite number,Y

The Y value is the highest coordinate on the screen that the sprite may occupy. Accepted values are 1–255.

Bottom Border

SYS 52843,sprite number,Y

Here, Y is the lowest coordinate on the screen that the sprite may occupy. Accepted values are, again, 1–255.

Left Border

SYS 52846,sprite number,X

The X value is the leftmost coordinate the sprite may occupy. Accepted values are 1–319.

Right Border

SYS 52849,sprite number,X

Here, X is the rightmost coordinate the sprite may occupy. Accepted values are, again, 1–319.

Wraparound

SYS 52852,sprite number,on/off

The Wraparound command makes a sprite reappear at the opposite border and continue moving in the same direction. Type 1 to turn wraparound on, 0 to turn it off.

Bouncing

SYS 52855,sprite number,on/off

The Bouncing command makes a sprite bounce off its border and move in the opposite direction. Type 1 to turn bouncing on, 0 to turn it off.

Wraparound dominates if both it and bouncing are activated for a sprite.

Animation

If you've ever tried animation on your own, you know it's a bother. You have to worry about continually calling your animation subroutine, and, if you're writing in Basic, you must worry about speed. Also, animation for several sprites at a time can slow program execution to a crawl.

Sprite Magician puts these problems in the past. Simply specify which sprite you're animating, the sprite block (0–255) where ani-

mation should start and the number of blocks it should continue. You can even specify the speed and whether or not to reset parameters.

SYS 52858,sprite number,starting block,number of blocks to animate,speed,reset

Acceptable values for the starting block (the number you'd normally poke into 2040–2047) are 0–255. Speed equals 60 divided by the number of blocks per second to be animated.

The reset parameter is optional. Normally, at the end of an animation sequence, the sequence is repeated in reverse. For example, the command SYS 52858,1,192,3,10 would initiate the block sequence 192,193,194,193,192,193,194. . .and so forth. However, with Sprite Magician, you can reset to the first value after each sequence. Continuing our example, instead of 192,193,194,193, 192, you'd have 192,193,194,192,193,194. A value of 1 for the reset parameter sets the flag to reset each time. If this parameter is left out, or you give it a value of 0, the sequence keeps reversing.

Motion Freezer

SYS 52861,sprite number,on/off

A value of 0 for the on/off parameter stops a sprite's motion; 1 starts it again.

Show Sprite

SYS 52864,sprite number,on/off

A value of 1 for the on/off parameter turns a sprite on and makes it visible; 0 makes it vanish.

Program Notes

Sprite Magician resides in memory locations 51456–52965 and uses 52966–53247 for miscellaneous tables. *Keep other code out of these areas.*

I tried to write Sprite Magician so it would work with other routines that change the IRQ vector. However, I can't guarantee it will work with any particular routine, so I advise you to activate Sprite Magician *after* another routine.

Because Sprite Magician does so much 60 times each second, I feared it might slow the C-64 down. However, after running several tests, I'm happy to say that the computer still zips right along.

RUN it right: **C-128; 1571 or 1581**

Device Toggler

By Mark Jordan

SOFTWARE SWITCHES CAN MAKE YOU a more efficient computer user. My short machine language program, Device Toggler, which I presented in *RUN*'s October 128 Mode column, switches devices 8 and 9 when you press shift/restore. This can be a big timesaver while programming, because you can keep two disks in place and grab data from either drive without having to type in those extra U commands that Basic requires.

To use the routine, load DEVICE.SWITCH, using Menu 128. Be aware, however, that some software will overwrite the interrupts involved.

RUN it right: **C-64**

Apple Harvest

By Joey Latimer

THE SCENE IS AN AUTUMN MORNING in 1636. You're a young pilgrim who's been sent out to collect apples for a Thanksgiving feast. The apples are so ripe that they're falling off the trees. Using your trusty basket, you run around and try to catch as many as you can.

To hone your apple-gathering skills, I've created Basket Case. In the game, you earn 100 points for each apple you catch, plus a bonus of 50 points per apple, credited to you after all the apples have fallen. If you're not nimble enough and get beaned by a falling apple, you lose your accumulated points and must start the game over at the beginning.

If you've looked in all your closets and still can't find your pilgrim

outfit, you can play Basket Case on your C-64. Just load it, using Menu 64, in 64 mode. Pressing B moves your on-screen pilgrim left; N moves the pilgrim right. Catch lots of apples, and you'll have a delicious Thanksgiving feast!

RUN it right: **C-64 or C-128; 1541 or 1571**

Disk Directory Organizer

By Jim Borden

I USE MY Five-Column Directory Listings program to keep track of all my disk directories. It works with any Star or Epson printer that allows for compressed type (a minimum of 16 cpi and 8 lines per inch). Other printers that meet these requirements will also work if the printer codes are put in the print subroutine starting at line 610. When you run my program, the directories are printed five columns wide with approximately 78 names in each column, or 370 files per page.

Five-Column Directory Listings is easy to use: Set your interface to ASCII (Commodore graphics in Compressed mode print too wide), align the paper at the top of the page and turn on the printer; then enter the name and date string. I use the name of the section of disks I'm working with, and the date the listing was made—for example, MUSIC PROGRAMS AND SONGS—10/05/90. Make sure you don't use any quotes, commas or colons.

Next you'll be prompted for a page number. I like to use ten times the section number in my disk box. For instance, my music programs are in section 4, so I use a page number of 40 to start the printout. This method groups all my music file directories with page numbers in the forties. Use a system that's convenient for you, but remember that this is a number and not a string.

The program is now ready to read the filenames for this section. You'll be asked if the directory you're about to read should be alphabetized. (I seldom use this option because it moves any dividers that are in the directory. However, it might make finding a program name easier.) The default answer is no.

The next prompt asks you to insert the disk. If you've read in other disks, the last disk's name appears to help you keep track. After you insert the disk, press the space bar to begin reading the directory into memory.

When the directory has been read (and sorted, if desired), the number of files on the disk is added to the Blocks Free line as a negative number. This helps you see if many short programs have filled the directory.

If there are more disks in the same section, insert the next disk at the prompt and press the space bar. Continue in this manner until all your disks have been read. If the page array should fill up before you read all the disks, the page will be printed and another one started in memory. When the section is done, enter N at the prompt for another disk, and any unfinished page will be printed.

To print another section, simply run the program again and enter the new name and date string along with a different beginning page number.

I find that this program prints directories in a compact form that's easy to update. I hope it works well for you, too. ■

EDITOR-IN-CHIEF
DENNIS BRISSON

TECHNICAL MANAGER
TIM WALSH

MANAGING EDITOR
BETH S. JALA

SENIOR EDITOR
HAROLD R. BJORNSEN

COPY EDITOR
PEG LEPAGE

ART DIRECTOR
HOWARD G. HAPP

DESIGN AND LAYOUT
ANN DILLON

TYPESETTING
DEBRA DAVIES
KEN SUTCLIFFE

FULFILLMENT CONSULTANT
DEBBIE BOURGAULT

11 Programs Included on this Disk

From the August/September RUN:

- How Far Is It To . . . ?
- Exercise Your Mouscles
- Pop-Top!
- Horseshoes

From the October RUN:

- A Notable Basic
- Time Clock
- RUN Paint Renamer
- Sprite Magician
- Device Toggler

Plus: Extra Bonus Programs!

- Apple Harvest
- Disk Directory Organizer

If any manufacturing defect becomes apparent, the defective disk will be replaced free of charge if returned by prepaid mail within 30 days of purchase. Send it, with a letter specifying the defect, to:

***RUN* Special Products • 80 Elm Street • Peterborough, NH 03458**

Replacements will not be made if the disk has been altered, repaired or misused through negligence, or if it shows signs of excessive wear or is damaged by equipment.

The programs in ReRUN are taken directly from listings prepared to accompany articles in *RUN* magazine. They will not run under all system configurations. Use the RUN It Right information included with each article as your guide.